Towards Traceability Metamodel for Business Process Modeling Notation

Saulius Pavalkis, Lina Nemuraite, Edita Milevičienė

Towards Traceability Metamodel for Business Process Modeling Notation

Saulius Pavalkis[1,2], Lina Nemuraite[1], Edita Mileviciene[2]

[1]Kaunas University of Technology, Department of Information Systems,
Studentu 50–308, LT-51368 Kaunas, Lithuania,
[2]No Magic Europe, Savanoriu av. 363, LT-49425 Kaunas, Lithuania,
`saulius.pavalkis@nomagic.com, lina.nemuraite@ktu.lt,`
`edita.mileviciene@nomagic.com,`

Abstract. This paper presents the traceability metamodel for Business Process Model and Notation (BPMN) and its implementation in Cameo Business Modeler plug-in for MagicDraw. There is no public standard traceability metamodel defined for BPMN yet. We present solutions that we have already applied in practice: we improve the traceability of BPMN models by defining derived properties that are calculated by a modeling tool on the fly. In contrast to other existing solutions, this approach does not require additional efforts from users for defining and maintaining traceability, and does not overload projects with redundant information. Using this approach, CASE tool developers are able to supplement their tools with traceability analysis means allowing users to access traceability information easier, to check completeness and correctness of BPMN models, and to analyze the impact of changes.

Keywords: traceability, derived properties, BPMN, model consistency, coverage analysis, change impact analysis.

1 Introduction

Today’s software is becoming more and more complex. Modeling takes an important role in the software development because of the ability to raise the level of abstraction from code to models using popular modeling languages such as UML [2], BPMN [1], SysML [3], and others. Models become primary artifacts in software and systems development. They cover all stages of software development from business analysis and requirements definition to implementation, code generation, and testing, as defined by Unified Software Development Process [4]. As a result, the complexity of models is growing, and this leads to increased risk and higher costs of software projects [5]. In this complex context, it becomes crucial to assure safety, reliability, and quality of software and systems granting their integrity, avoiding redundancy, managing development processes and changes. Model traceability can help to reach these goals since it is able to reduce complexity by easing comprehension of design decisions to stakeholders, decision makers, and developers.

Traceability is the important aspect of software development for analyzing risks and costs during change management. On the other hand, BPMN (we are concerning the second BPMN version, BPMN 2) is one of the most popular standards for business process modeling. Many modeling tools support BPMN diagrams [6]. However, traceability of important elements of BPMN models is not assured.

In our viewpoint, the traceability information should be created, updated and visualized in such a way that it would not cause more problems than advantages received. It should not unpredictably increase the overhead and costs of the project. This is especially important for business users (i.e. main users of BPMN) working with visual representation of models. Traceability information should be presented in a clear and comprehensive way in order to be understood and accepted by business process modelers.

The traceability of BPMN models can be improved by using our proposed derived property approach. The core of the approach is an extension of a metamodel of the problematic modeling language with additional properties that can be calculated by a modeling tool on the fly. In contrast to other existing solutions, this approach does not require users defining and maintaining traceability relations in their projects, and does not overload their projects with traceability information. Using this approach, CASE tool developers are able to supplement their tools with traceability analysis means allowing users to access traceability information easier, to check completeness and correctness of BPMN models, and to analyze impact of changes.

The rest of the paper is structured as follows. Section 2 analyzes related works. Section 3 presents the traceability metamodel for BPMN and its implementation in Cameo Business Modeler tool using custom derived properties dedicated for traceability. Finally, section 4 presents conclusions and future works.

2 Traceability Concepts and Related Works

In the IEEE Standard Glossary of Software Engineering Terminology [7], the traceability is defined as “The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor–successor or master–subordinate relationship to one another”. In order words, traceability is understood as the ability to identify direct or indirect relations among project elements. There are many other traceability definitions; however, most of them are similar to the presented ones.

Traceability is classified in different ways on the base of various aspects. According to [8], there are some fundamental classifications, like forward [9], backwards [10], horizontal [11, 12], and vertical traceability [11].

Ramesh and Edwards [11] define the distinction between horizontal and vertical traceability. Traceability that considers links among artifacts belonging to the same project phase or level of abstraction is defined as horizontal traceability; traceability that links artifacts belonging to different phases or levels is defined by vertical traceability. In this paper, we will focus on horizontal traceability links within BPMN models.

Traceability schema, or metamodel of a particular domain defines what relations between specific model elements are treated as traceability relations, and what semantics they carry. Multiple authors have proposed traceability metamodels [13−16] but no common understanding of a complete traceability metamodel [9] is defined yet. The limitation of these approaches arises because of inflexible types of relationships while needs and practices of organizations are changing. A relevant solution should provide traceability metamodel, which supports customization and extensibility of traceability links giving possibility to define new types of links, artifacts, and transitive relations. Such capabilities and predefined schemas are provided in our derived properties based approach which is already applied to UML and SysML.

BPMN role in software development is discussed in multiple sources [17]. However, there is no standard traceability metamodel for BPMN. The lack of traceability in BPMN models causes a number of problems:

1st problem. Resource roles take part in BPMN activities that belong to some process. However, there is no direct relation between resource role and process in BPMN metamodel. This means it is impossible to trace information about processes in which the resource role takes part. For solving this problem, we introduce a traceability rule defining relation between resource role and process, and vice versa.

2nd problem. Identification of business concepts is crucial for business process modeling, and we apply UML class diagram for this purpose [17]. Instances of concepts taking part in workflows of business process models are represented as objects. However, it is not possible to find out in which processes business concepts take part. We propose a traceability rule defining relation between business concept (class or BPMN resource) and processes, in which it takes part, and vice versa.

3rd problem. Participants take part in sending and receiving messages during process execution. In BPMN metamodel, there is no direct relation between participants and messages sent or received by them. Consequently, we introduce a traceability rule defining messages sent and received by participants.

One of the most important aspects of traceability analysis tools is their ability to represent results. Winkler et al. emphasizes matrices, cross-references, and graph-based representations as main methods for traceability visualization [9], and summarizes other traceability benefits: prioritizing requirements, estimating change impact, proving system adequateness, understanding the system, supporting design decisions, validating and much more.

Cameo Business Modeler covers methods for traceability visualization together with other visualization and analysis means. Also, we provide other capabilities for analyzing models on the base of traceability information: change impact analysis [18], checking consistency and completeness of models.

3 Traceability Metamodel for Business Process Model and Notation

We solve BPMN traceability problems by applying custom derived properties approach which allows to extend UML metamodel for derived properties specification and enables its customization as a part of MagicDraw DSL engine – a core of Cameo Business Modeler [19]. As the mentioned BPMN traceability problems are caused by absence of direct relations between elements, we use derived properties for creating such relations. Definition and calculation of derived properties are presented in section 3.1 "Derived Properties Framework and Metamodel" (full description may be found in [20]).

We define BPMN traceability rule expressions as property chains (Table 1) where column "Rule name" shows the name of the derived property; "Source element" identifies the owner of that derived property; "Target element" corresponds to a value of the derived property.

Table 1. BPMN traceability rules solving traceability problems

Rule name	Source element	Target element	Description
Resource Role – BPMN Process traceability			
1. Taking Part in Process	Resource Role	BPMN Process	Defines processes, in which activity resource role takes part.
2. Activity Resources	BPMN Process	Resource Role	Defines resources roles taking part in process activities.
Business Concept – BPMN Process			
3. Taking Part in Process	Class or BPMN resource	BPMN Process	Defines processes, in which business concepts takes part.
4. Business concepts	BPMN Process	Class or BPMN resource	Defines business concepts, which take part in business process.
Participant – Message			
5. Sent Messages	Participant	Message	Participants take part in sending and receiving messages during process execution. Property defines messages sent by participant
6. Received Messages	Participant	Message	Participants take part in sending and receiving messages during process execution. Property defines messages received by participant via message flow from representing pools.

3.1 Derived Properties Framework and Metamodel

In order to be able to define derived properties in modeling environment we extended a modeling language for specifying derived property details. UML and other modeling languages provided by Object Management Group (OMG) have the standard extension mechanism – profiling.

UML extension for derived properties reuses UML properties and MagicDraw DSL engine [19] constructs; it introduces only one stereotype with a single property for specifying derived properties.

The stereotype DerivedPropertySpecification extends UML metaclass Property for specification of derived property expression that defines how this property is calculated. UML properties of stereotyped property element are used for specifying a derived property as well: one can define the name, type, multiplicity, isUnique, isOrdered, isDerived, isReadOnly, and body of comment.

Stereotyped property with DerivedPropertySpecification stereotype is added into MagicDraw DSL [19] customization class (i.e. class stereotyped as customization). The definition of tag customizationTarget of this class specifies in which element type (UML or extended one) derived property will be created (for more details of specifying derived properties please refer to [20]).

The heart of the derived property is the expression according to which it is calculated. Several different types of such expressions are available in Cameo Business Modeler: simple – through UML properties and relationships; more advanced ones use OCL expressions, property chains, scripting languages or Java code.

Most popular are simple expressions and property chains. Property chain expression type (i.e. path through metamodel and properties) is used for navigating from a context element to a final linked property for gathering the resulting elements as derived property values. A property chain expression defines navigation path consisting of metaclass/stereotype and property/tag pairs. For example, for derived property "Taking Part in Process" (Fig. 2), the property chain expression is ResourceRole.opposite(BPMNActivity.resources).owner [20]. For derived property "Received Messages" (Fig. 8) property chain expression is Participant. opposite(ActivityPartition.represents).opposite(DirectedRelationship.source). messageRef [20].

Derived property expressions could be written in OCL. For example, derived property "Taking Part in Process" expression could by written in OCL as

```
context ResourceRole::takingPartInProcess:BPMNProcess
  derive: self.BPMNActivity.owner,
```

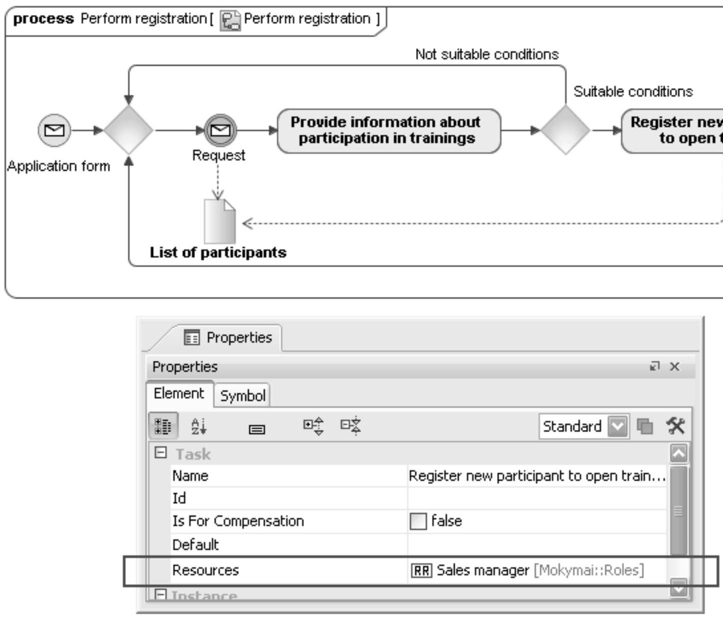
and "Received Messages" could be written in OCL as

```
context Participant::receivedMessage:Bag(Message)
  derive:self.activityPartition.directedRelationship→
    select(n| n.oclIsKindOf(MessageFlow)).messageRef
```

However, our current traceability implementation in MagicDraw allows users to specify property chains in a simpler way by choosing required elements and properties in dialog. OCL expressions could be applied for more complex situations.

3.2 Traceability Rules

Resource Roles taking part in BPMN activities (tasks and subprocesses) are shown in Fig. 1. BPMN standard property – Resources shows all resource roles taking part in a particular activity.


process Perform registration [Perform registration]
Not suitable conditions
Suitable conditions
Application form
Request
Provide information about participation in trainings
Register new participant to open trainings
List of participants
Properties
Element
Symbol
Standard

⊟ Task	
Name	Register new participant to open train…
Id	
Is For Compensation	☐ false
Default	
Resources	RR Sales manager [Mokymai::Roles]
⊟ Instance	

Fig. 1. BPMN process diagram representing registration for open training class

For assuring Resource Role – Process traceability, we introduce bidirectional traceability relation between resource role and BPMN process **(**Fig. 2**).** In result, we can see all resources, participating in activities of a particular BPMN process, in the specification of that process (Fig. 3).

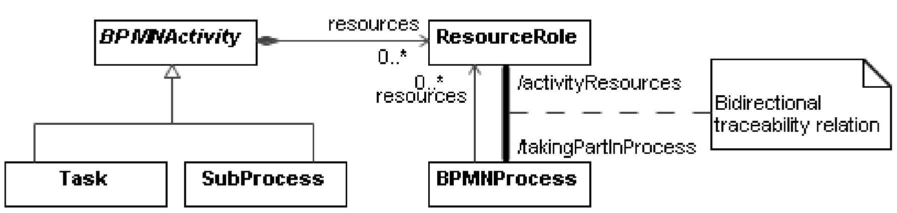
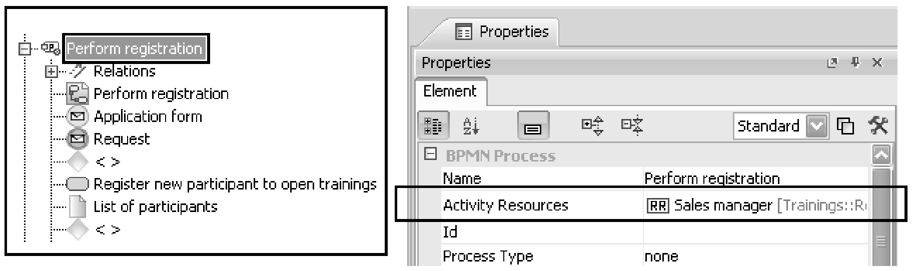


Fig. 2. BPMN metamodel extension for traceability relation between resource role and BPMN process


Perform registration
Relations
Perform registration
Application form
Request
< >
Register new participant to open trainings
List of participants
< >

Properties

BPMN Process	
Name	Perform registration
Activity Resources	Sales manager [Trainings::R
Id	
Process Type	none

Fig. 3. Derived property “Activity Resources” in the BPMN process specification

Definition of business processes can start from identifying business concepts. For this purpose, we can use UML class diagram [17]. Business concepts (classes or BPMN resources) taking part in the workflows of business processes are represented as data objects (Fig. 4). However, there is no possibility to trace from business concepts to processes where instances of concepts (data objects) are used.

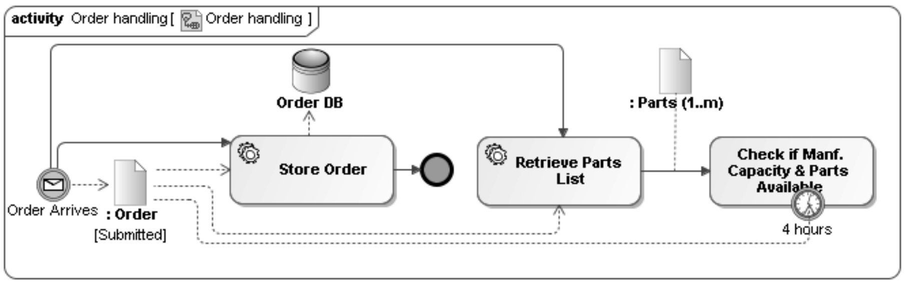


Fig. 4. BPMN diagram for Order Handling Process

Metamodel for solving Business Concept – BPMN Process traceability problem is presented in Fig. 5. This metamodel introduces a relation between business concept (class or BPMN resource) and process in which it takes part.

This extension allows to show all business concepts, taking part in BPMN process, in MagicDraw Dependency Matrix (Fig. 6). Resource which is not used as business concept will still have traceability property, but it will not have a value pointing to BPMN process.

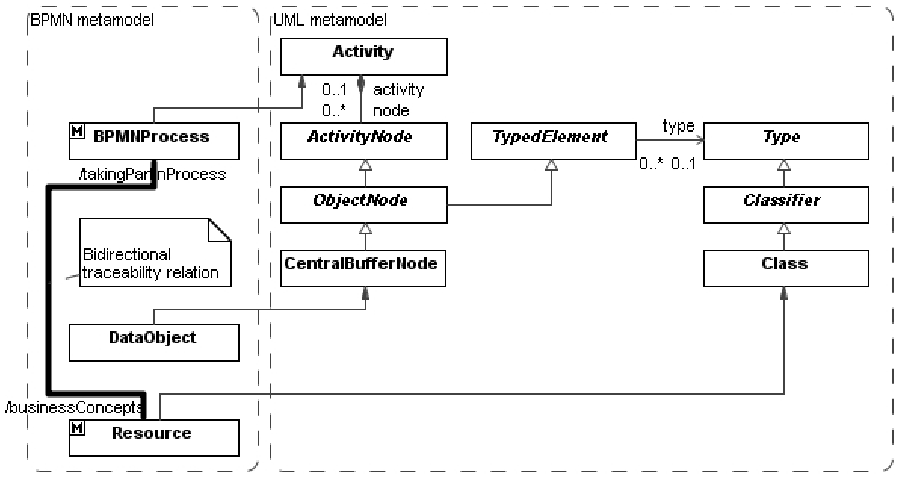
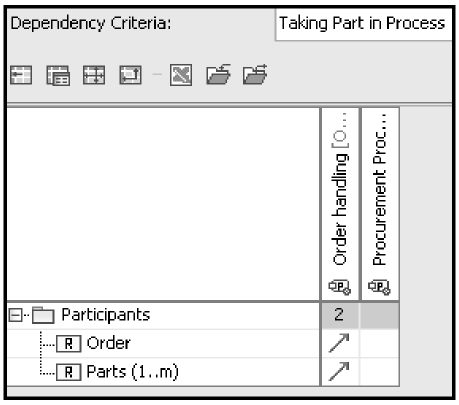


Fig. 5. Parts of BPMN and UML metamodels extended with traceability relation between Resource and BPMN process



Dependency Criteria: Taking Part in Process

	Order handling [O...	Procurement Proc...
Participants	2	
Order	↗	
Parts (1..m)	↗	

Fig. 6. Visualization of traceability property "Taking Part in Process" with Dependency Matrix

Similarly, BPMN collaboration diagram is the only place where participants and messages sent or received by them can be seen (Fig. 7). There are no direct relation between participant and message in BPMN metamodel. For assuring this kind of traceability, we introduce two traceability relations between Participant and Message (Fig. 8). These relations allow to show messages of a particular participant in its specification (Fig. 9).

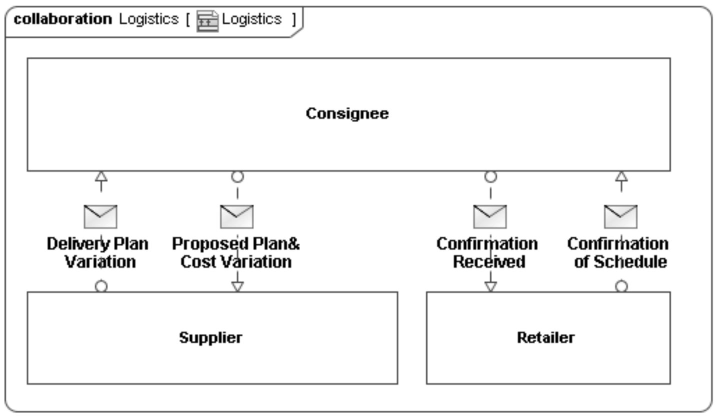


Fig. 7. BPMN Collaboration diagram showing messages sent between participants

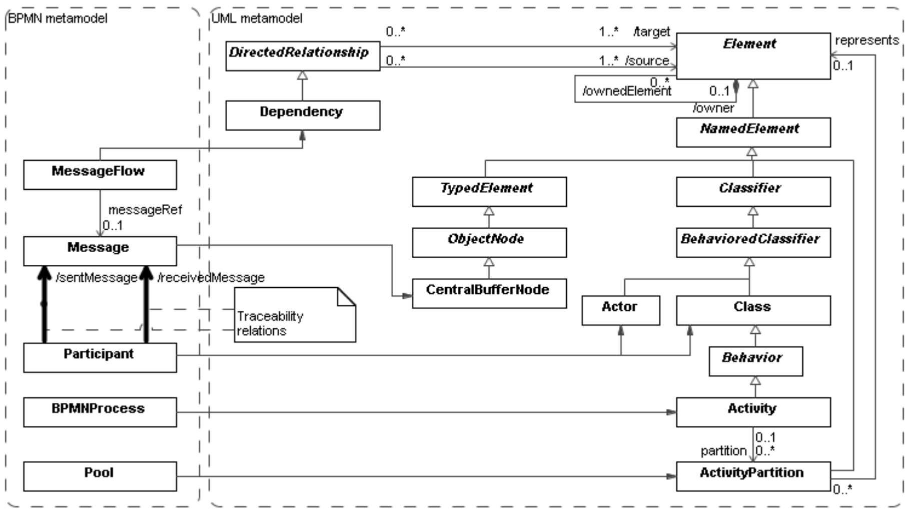
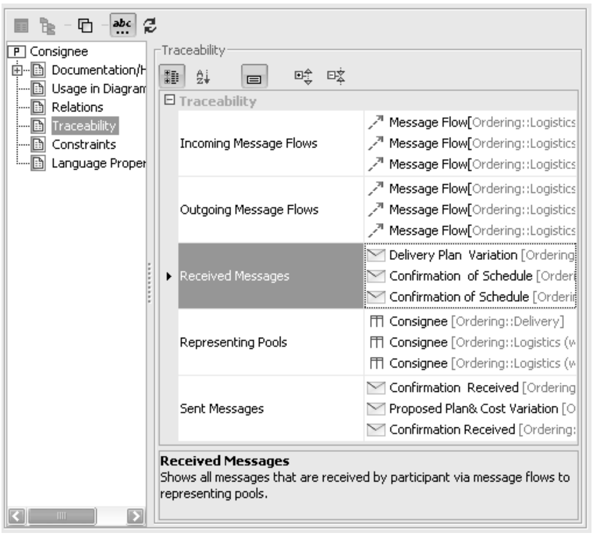


Fig. 8. Parts of BPMN and UML metamodels extended with traceability relation between participants and messages sent or received by the participant



Traceability	
Incoming Message Flows	Message Flow[Ordering::Logistics Message Flow[Ordering::Logistics Message Flow[Ordering::Logistics
Outgoing Message Flows	Message Flow[Ordering::Logistics Message Flow[Ordering::Logistics Message Flow[Ordering::Logistics
Received Messages	Delivery Plan Variation [Ordering Confirmation of Schedule [Order Confirmation of Schedule [Orderi
Representing Pools	Consignee [Ordering::Delivery] Consignee [Ordering::Logistics (w Consignee [Ordering::Logistics (w
Sent Messages	Confirmation Received [Ordering Proposed Plan& Cost Variation [O Confirmation Received [Ordering:

Received Messages
Shows all messages that are received by participant via message flows to representing pools.

Fig. 9. Participant specification extended with derived traceability properties – Sent Messages and Received Messages

3.3 Visualization and Analysis of Traceability Rules

Once derived properties are specified, they appear in specification dialog of the corresponding elements and other places in the same way as regular BPMN properties. Now, by visualizing and analyzing traceability information, we can discover related elements, which will be impacted by model changes (i.e. we can perform impact analysis). Impact analysis is performed by discovering impacted parts – the ones related with traceability relations. The following paragraph overviews methods for discovery of impacted parts. Also, we can validate model consistency by performing coverage analysis for discovering whether all requirements are satisfied by design and verified with test cases, or not.

Transitive traceability can be visualized by Relation Map – a graph based visualization that allows review and analysis of multilevel relations. Dependency Matrix represents traceability relations between requirements and design**.** Traceability properties also can be visualized on diagram using standard MagicDraw mechanism for displaying property values in notes. Traceability property groups can be seen in the contextual menu of an element.

Generic table is an easy way for performing coverage analysis e.g. empty cells in the rows indicate a lack of consistency in the model. Coverage analysis report can be generated using documentation generation capability. All the derived properties, together with custom BPMN properties, can be accessed when creating user specific report templates.

Finally, you can check completeness of traceability and validate non-existence of cyclic relationships by using CBM validation feature. Predefined validation suite for traceability checks model for empty traceability properties and elements involved in both forward and backward traceability relations with another elements.

4 Conclusions and Future Works

1. Analysis of traceability in current BPMN 2 models had shown the lack of traceability between BPMN concepts, from which we first had taken into consideration processes and resource roles, BPMN processes and business concepts, participants and messages as most required ones.
2. As the solution for these problems, we propose traceability metamodel based on UML derived properties. Our proposed derived property approach is already tried in practice for UML and SysML, and could be adopted by other developers in other tools and for other languages as currently we have done for BPMN.
3. In contrast to other proposals, derived property based traceability framework supports customization and extensibility of traceability links giving possibility to define new types of links, artifacts, and transitive relations. Also, it has advantages with significant decrease of overhead as derived properties are automatically calculated by a modeling tool and dynamically updated according to changes in models. No manual work is required to specify traceability information – it is created and updated fully automatically.
4. Implementation of traceability metamodel in Cameo Business Modeler (CBM) allows reusing existing CBM means as dependency matrices, report templates, and validation rules for traceability information analysis, visualization and navigation.
5. Proposed BPMN 2 traceability metamodel and rules provide information about dependencies between BPMN processes and resource roles, processes and business concepts, participants and messages. This allows validating BPMN 2 models for correctness and completeness of these aspects, and analyzing impact of changes.
6. In our future work, we will concentrate on extending the traceability metamodel for BPMN 2 as well as defining specific traceability metamodels for modeling databases and enterprise architectures.

Acknowledgement. The authors would like to thank No Magic, Inc, especially the MagicDraw and Cameo Business Modeler product teams for comprehensive support.

References

1. OMG: OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.3. OMG Document Number: formal/2010-05-05 (2010)
2. OMG: Business Process Model and Notation (BPMN), Version 2.0. OMG Document Number: formal/2011-01-03 (2010)
3. OMG: OMG Systems Modeling Language (OMG SysML), Version 1.2. OMG Document

Number: formal/2010-06-01 (2010)

4. Jacobson, I., Booch, G., Rumbaugh, J.: The Unified Software Development Process. Addison-Wesley Professional, Boston, MA (1999)
5. Biffl, S., Aurum, A., Boehm, B., Erdogmus, H., Grünbacher, P.: Value-based software engineering. Springer-Verlag New York, Inc. Secaucus, NJ, USA (2005)
6. Allweyer, T.:BPMN 2.0 – Introduction to the Standard for Business Process Modeling. Books on Demand GmbH (2010)
7. IEEE Standards Board.: IEEE Standard Glossary of Software Engineering Terminology, IEEE Std 610.12-1990. IEEE Press, Piscataway (1990)
8. Aizenbud-Reshef, N., Nolan, B.T., Rubin, J., Shaham-Gafni, Y.: Model traceability. IBM Systems Journal, 45(3), 515–526 (2006)
9. Winkler, S., Pilgrim, J.: A survey of traceability in requirements engineering and model-driven development. Software and Systems Modeling, 9(4), 529–565 (2010)
10. IEEE Standards Board: IEEE Guide to Software Requirements Specification, ANSI/IEEE Std 830-1984. IEEE Press, Piscataway (1984)
11. Ramesh, B., Edwards, M.: Issues in the development of a requirements traceability model. In: Proceedings of the IEEE International Symposium on Requirements Engineering, pp. 256–259. IEEE Computer Society, New York (1993)
12. Briand, L. C., Labiche, Y., Yue, T.,: Automated traceability analysis for UML model refinements. Information and Software Technology, 51, 512–527 (2009)
13. Ramesh, B., Jarke, M.: Toward reference models for requirements traceability. IEEE Transactions on Software Engineering, 27(1), 58–93 (2001)
14. Pinheiro, F. A. C., Goguen, J. A.: An Object-Oriented Tool for Tracing Requirements. IEEE Software, 13(2), 52–64 (1996)
15. Dick, J.: Rich Traceability. In: Proceedings of the 1st International Workshop on Traceability in Emerging Forms of Software Engineering, pp. 18–23, Edinburgh, Scotland (2002)
16. Letelier, P.: A Framework for Requirements Traceability in UML-Based Projects. In: Proceedings of the 1st International Workshop on Traceability in Emerging Forms of Software Engineering, pp. 30–41, Edinburgh, Scotland (2002)
17. Silingas D., Butleris R.: Towards implementing a framework for modeling software requirements in MagicDraw UML. Information Technology and Control, 38(2), 153–164. (2009)
18. Crnković, I., Asklund, U., Persson-Dahlqvist, A.: Implementing and Integrating Product Data Management and Software Configuration Management. Artech House, London (2003)
19. Silingas, D., Vitiutinas, R., Armonas, A., Nemuraite, L:. Domain-specific modeling environment based on UML profiles. In: Information Technologies' 2009: proceedings of the 15th International Conference on Information and Software Technologies, IT 2009, Kaunas, Lithuania, April 23-24, 2009, pp. 167–177. Kaunas University of Technology, Technologija, Kaunas (2009)
20. UML Profiling and DSL User Guide, https://secure.nomagic.com/files/manuals/UML%20Profiling%20and%20DSL%20UserGuide.pdf